
Subject: [RFC][PATCH 6/7] UBC: kernel memory accounting (mark objects)

Posted by [dev](#) on Wed, 16 Aug 2006 15:40:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

Mark some kmem caches with SLAB_UBC and some allocations with __GFP_UBC to cause charging/limiting of appropriate kernel resources.

Signed-Off-By: Pavel Emelianov <xemul@sw.ru>

Signed-Off-By: Kirill Korotaev <dev@sw.ru>

```
---
arch/i386/kernel/ldt.c      |  4 ++--
arch/i386/mm/init.c         |  4 ++--
arch/i386/mm/pgtable.c      |  6 +++++-
drivers/char/tty_io.c       | 10 ++++++----
fs/file.c                   |  8 +++++----
fs/locks.c                  |  2 +-
fs/namespace.c              |  3 ++-
fs/select.c                 |  7 +++++---
include/asm-i386/thread_info.h |  4 ++--
include/asm-ia64/pgalloc.h  | 24 ++++++-----
include/asm-x86_64/pgalloc.h | 12 ++++++----
include/asm-x86_64/thread_info.h |  5 +++--
ipc/msgutil.c               |  4 ++--
ipc/sem.c                   |  7 +++++---
ipc/util.c                  |  8 +++++----
kernel/fork.c                | 15 ++++++-----
kernel/posix-timers.c       |  3 ++-
kernel/signal.c             |  2 +-
kernel/user.c               |  2 +-
mm/rmap.c                   |  3 ++-
mm/shmem.c                  |  3 ++-
21 files changed, 80 insertions(+), 56 deletions(-)
```

```
--- ./arch/i386/kernel/ldt.c.ubslabs 2006-04-21 11:59:31.000000000 +0400
```

```
+++ ./arch/i386/kernel/ldt.c 2006-08-01 13:22:30.000000000 +0400
```

```
@@ -39,9 +39,9 @@ static int alloc_ldt(mm_context_t *pc, i
    oldsize = pc->size;
    mincount = (mincount+511)&(~511);
    if (mincount*LDT_ENTRY_SIZE > PAGE_SIZE)
-   newldt = vmalloc(mincount*LDT_ENTRY_SIZE);
+   newldt = vmalloc_ub(mincount*LDT_ENTRY_SIZE);
    else
-   newldt = kmalloc(mincount*LDT_ENTRY_SIZE, GFP_KERNEL);
+   newldt = kmalloc(mincount*LDT_ENTRY_SIZE, GFP_KERNEL_UBC);

    if (!newldt)
        return -ENOMEM;
```

```

--- ./arch/i386/mm/init.c.ubslabs 2006-07-10 12:39:10.000000000 +0400
+++ ./arch/i386/mm/init.c 2006-08-01 13:17:07.000000000 +0400
@@ -680,7 +680,7 @@ void __init pgtable_cache_init(void)
    pmd_cache = kmem_cache_create("pmd",
        PTRS_PER_PMD*sizeof(pmd_t),
        PTRS_PER_PMD*sizeof(pmd_t),
-    0,
+    SLAB_UBC,
        pmd_ctor,
        NULL);
    if (!pmd_cache)
@@ -689,7 +689,7 @@ void __init pgtable_cache_init(void)
    pgd_cache = kmem_cache_create("pgd",
        PTRS_PER_PGD*sizeof(pgd_t),
        PTRS_PER_PGD*sizeof(pgd_t),
-    0,
+    SLAB_UBC,
        pgd_ctor,
        PTRS_PER_PMD == 1 ? pgd_dtor : NULL);
    if (!pgd_cache)
--- ./arch/i386/mm/pgtable.c.ubslabs 2006-07-10 12:39:10.000000000 +0400
+++ ./arch/i386/mm/pgtable.c 2006-08-01 13:27:35.000000000 +0400
@@ -158,9 +158,11 @@ struct page *pte_alloc_one(struct mm_str
    struct page *pte;

#ifdef CONFIG_HIGHPTE
-    pte = alloc_pages(GFP_KERNEL|__GFP_HIGHMEM|__GFP_REPEAT|__GFP_ZERO, 0);
+    pte = alloc_pages(GFP_KERNEL|__GFP_HIGHMEM|__GFP_REPEAT|__GFP_ZERO |
+    __GFP_UBC | __GFP_UBC_LIMIT, 0);
#else
-    pte = alloc_pages(GFP_KERNEL|__GFP_REPEAT|__GFP_ZERO, 0);
+    pte = alloc_pages(GFP_KERNEL|__GFP_REPEAT|__GFP_ZERO|
+    __GFP_UBC | __GFP_UBC_LIMIT, 0);
#endif
    return pte;
}
--- ./drivers/char/tty_io.c.ubslabs 2006-07-10 12:39:11.000000000 +0400
+++ ./drivers/char/tty_io.c 2006-08-01 15:21:21.000000000 +0400
@@ -158,7 +158,7 @@ static struct tty_struct *alloc_tty_stru
{
    struct tty_struct *tty;

-    tty = kmalloc(sizeof(struct tty_struct), GFP_KERNEL);
+    tty = kmalloc(sizeof(struct tty_struct), GFP_KERNEL_UBC);
    if (tty)
        memset(tty, 0, sizeof(struct tty_struct));
    return tty;
@@ -1495,7 +1495,7 @@ static int init_dev(struct tty_driver *d

```

```

if (!*tp_loc) {
    tp = (struct termios *) kmalloc(sizeof(struct termios),
-   GFP_KERNEL);
+   GFP_KERNEL_UBC);
    if (!tp)
        goto free_mem_out;
    *tp = driver->init_termios;
@@ -1503,7 +1503,7 @@ static int init_dev(struct tty_driver *d

```

```

if (!*ltp_loc) {
    ltp = (struct termios *) kmalloc(sizeof(struct termios),
-   GFP_KERNEL);
+   GFP_KERNEL_UBC);
    if (!ltp)
        goto free_mem_out;
    memset(ltp, 0, sizeof(struct termios));
@@ -1528,7 +1528,7 @@ static int init_dev(struct tty_driver *d

```

```

if (!*o_tp_loc) {
    o_tp = (struct termios *)
-   kmalloc(sizeof(struct termios), GFP_KERNEL);
+   kmalloc(sizeof(struct termios), GFP_KERNEL_UBC);
    if (!o_tp)
        goto free_mem_out;
    *o_tp = driver->other->init_termios;
@@ -1536,7 +1536,7 @@ static int init_dev(struct tty_driver *d

```

```

if (!*o_ltp_loc) {
    o_ltp = (struct termios *)
-   kmalloc(sizeof(struct termios), GFP_KERNEL);
+   kmalloc(sizeof(struct termios), GFP_KERNEL_UBC);
    if (!o_ltp)
        goto free_mem_out;
    memset(o_ltp, 0, sizeof(struct termios));
--- ./fs/file.c.ubslabs 2006-07-17 17:01:12.000000000 +0400
+++ ./fs/file.c 2006-08-01 15:18:03.000000000 +0400
@@ -44,9 +44,9 @@ struct file ** alloc_fd_array(int num)
    int size = num * sizeof(struct file *);

```

```

if (size <= PAGE_SIZE)
-   new_fds = (struct file **) kmalloc(size, GFP_KERNEL);
+   new_fds = (struct file **) kmalloc(size, GFP_KERNEL_UBC);
else
-   new_fds = (struct file **) vmalloc(size);
+   new_fds = (struct file **) vmalloc_ub(size);
    return new_fds;
}

```

```

@@ -213,9 +213,9 @@ fd_set * alloc_fdset(int num)
    int size = num / 8;

    if (size <= PAGE_SIZE)
-   new_fdset = (fd_set *) kmalloc(size, GFP_KERNEL);
+   new_fdset = (fd_set *) kmalloc(size, GFP_KERNEL_UBC);
    else
-   new_fdset = (fd_set *) vmalloc(size);
+   new_fdset = (fd_set *) vmalloc_ub(size);
    return new_fdset;
}

--- ./fs/locks.c.ubslabs 2006-07-10 12:39:16.000000000 +0400
+++ ./fs/locks.c 2006-08-01 12:46:47.000000000 +0400
@@ -2226,7 +2226,7 @@ EXPORT_SYMBOL(lock_may_write);
static int __init filelock_init(void)
{
    filelock_cache = kmem_cache_create("file_lock_cache",
-   sizeof(struct file_lock), 0, SLAB_PANIC,
+   sizeof(struct file_lock), 0, SLAB_PANIC | SLAB_UBC,
    init_once, NULL);
    return 0;
}

--- ./fs/namespace.c.ubslabs 2006-07-10 12:39:16.000000000 +0400
+++ ./fs/namespace.c 2006-08-01 12:47:12.000000000 +0400
@@ -1825,7 +1825,8 @@ void __init mnt_init(unsigned long mempa
    init_rwsem(&namespace_sem);

    mnt_cache = kmem_cache_create("mnt_cache", sizeof(struct vfsmount),
-   0, SLAB_HWCACHE_ALIGN | SLAB_PANIC, NULL, NULL);
+   0, SLAB_HWCACHE_ALIGN | SLAB_UBC | SLAB_PANIC,
+   NULL, NULL);

    mount_hashtable = (struct list_head *)__get_free_page(GFP_ATOMIC);

--- ./fs/select.c.ubslabs 2006-07-10 12:39:17.000000000 +0400
+++ ./fs/select.c 2006-08-01 15:17:01.000000000 +0400
@@ -103,7 +103,8 @@ static struct poll_table_entry *poll_get
    if (!table || POLL_TABLE_FULL(table)) {
        struct poll_table_page *new_table;

-   new_table = (struct poll_table_page *) __get_free_page(GFP_KERNEL);
+   new_table = (struct poll_table_page *)
+   __get_free_page(GFP_KERNEL_UBC);
        if (!new_table) {
            p->error = -ENOMEM;
            __set_current_state(TASK_RUNNING);

```

```

@@ -339,7 +340,7 @@ static int core_sys_select(int n, fd_set
if (size > sizeof(stack_fds) / 6) {
/* Not enough space in on-stack array; must use kmalloc */
ret = -ENOMEM;
- bits = kmalloc(6 * size, GFP_KERNEL);
+ bits = kmalloc(6 * size, GFP_KERNEL_UBC);
if (!bits)
goto out_nofds;
}
@@ -693,7 +694,7 @@ int do_sys_poll(struct pollfd __user *uf
if (!stack_pp)
stack_pp = pp = (struct poll_list *)stack_pps;
else {
- pp = kmalloc(size, GFP_KERNEL);
+ pp = kmalloc(size, GFP_KERNEL_UBC);
if (!pp)
goto out_fds;
}
--- ./include/asm-i386/thread_info.h.ubslabs 2006-07-10 12:39:19.000000000 +0400
+++ ./include/asm-i386/thread_info.h 2006-08-01 15:19:50.000000000 +0400
@@ -99,13 +99,13 @@ static inline struct thread_info *current
({
\
struct thread_info *ret; \
\
- ret = kmalloc(THREAD_SIZE, GFP_KERNEL); \
+ ret = kmalloc(THREAD_SIZE, GFP_KERNEL_UBC); \
if (ret) \
memset(ret, 0, THREAD_SIZE); \
ret; \
})
#else
-#define alloc_thread_info(tsk) kmalloc(THREAD_SIZE, GFP_KERNEL)
+#define alloc_thread_info(tsk) kmalloc(THREAD_SIZE, GFP_KERNEL_UBC)
#endif

#define free_thread_info(info) kfree(info)
--- ./include/asm-ia64/pgalloc.h.ubslabs 2006-07-10 12:39:19.000000000 +0400
+++ ./include/asm-ia64/pgalloc.h 2006-08-01 13:35:49.000000000 +0400
@@ -19,6 +19,8 @@
#include <linux/page-flags.h>
#include <linux/threads.h>

+#include <ub/kmem.h>
+
#include <asm/mmu_context.h>

DECLARE_PER_CPU(unsigned long *, __pgtable_quicklist);
@@ -37,7 +39,7 @@ static inline long pgtable_quicklist_tot

```

```

    return ql_size;
}

```

```

-static inline void *pgtable_quicklist_alloc(void)
+static inline void *pgtable_quicklist_alloc(int charge)
{
    unsigned long *ret = NULL;

```

```

@@ -45,13 +47,20 @@ static inline void *pgtable_quicklist_al

```

```

    ret = pgtable_quicklist;
    if (likely(ret != NULL)) {
+ if (charge && ub_page_charge(virt_to_page(ret),
+ 0, __GFP_UBC_LIMIT)) {
+ ret = NULL;
+ goto out;
+ }
    pgtable_quicklist = (unsigned long *)(*ret);
    ret[0] = 0;
    --pgtable_quicklist_size;
+out:
    preempt_enable();
    } else {
    preempt_enable();
- ret = (unsigned long *)__get_free_page(GFP_KERNEL | __GFP_ZERO);
+ ret = (unsigned long *)__get_free_page(GFP_KERNEL |
+ __GFP_ZERO | __GFP_UBC | __GFP_UBC_LIMIT);
    }

```

```

    return ret;
@@ -69,6 +78,7 @@ static inline void pgtable_quicklist_fre
#endif

```

```

    preempt_disable();
+ ub_page_uncharge(virt_to_page(pgtable_entry), 0);
    *(unsigned long *)pgtable_entry = (unsigned long)pgtable_quicklist;
    pgtable_quicklist = (unsigned long *)pgtable_entry;
    ++pgtable_quicklist_size;

```

```

@@ -77,7 +87,7 @@ static inline void pgtable_quicklist_fre

```

```

static inline pgd_t *pgd_alloc(struct mm_struct *mm)
{
- return pgtable_quicklist_alloc();
+ return pgtable_quicklist_alloc(1);
}

```

```

static inline void pgd_free(pgd_t *pgd)
@@ -94,7 +104,7 @@ pgd_populate(struct mm_struct *mm, pgd_t

```

```

static inline pud_t *pud_alloc_one(struct mm_struct *mm, unsigned long addr)
{
- return pgtable_quicklist_alloc();
+ return pgtable_quicklist_alloc(1);
}

static inline void pud_free(pud_t * pud)
@@ -112,7 +122,7 @@ pud_populate(struct mm_struct *mm, pud_t

static inline pmd_t *pmd_alloc_one(struct mm_struct *mm, unsigned long addr)
{
- return pgtable_quicklist_alloc();
+ return pgtable_quicklist_alloc(1);
}

static inline void pmd_free(pmd_t * pmd)
@@ -137,13 +147,13 @@ pmd_populate_kernel(struct mm_struct *mm
static inline struct page *pte_alloc_one(struct mm_struct *mm,
    unsigned long addr)
{
- return virt_to_page(pgtable_quicklist_alloc());
+ return virt_to_page(pgtable_quicklist_alloc(1));
}

static inline pte_t *pte_alloc_one_kernel(struct mm_struct *mm,
    unsigned long addr)
{
- return pgtable_quicklist_alloc();
+ return pgtable_quicklist_alloc(0);
}

static inline void pte_free(struct page *pte)
--- ./include/asm-x86_64/pgalloc.h.ubslabs 2006-04-21 11:59:36.000000000 +0400
+++ ./include/asm-x86_64/pgalloc.h 2006-08-01 13:30:46.000000000 +0400
@@ -31,12 +31,14 @@ static inline void pmd_free(pmd_t *pmd)

static inline pmd_t *pmd_alloc_one (struct mm_struct *mm, unsigned long addr)
{
- return (pmd_t *)get_zeroed_page(GFP_KERNEL|__GFP_REPEAT);
+ return (pmd_t *)get_zeroed_page(GFP_KERNEL|__GFP_REPEAT|
+ __GFP_UBC | __GFP_UBC_LIMIT);
}

static inline pud_t *pud_alloc_one(struct mm_struct *mm, unsigned long addr)
{
- return (pud_t *)get_zeroed_page(GFP_KERNEL|__GFP_REPEAT);
+ return (pud_t *)get_zeroed_page(GFP_KERNEL|__GFP_REPEAT|

```

```

+ __GFP_UBC | __GFP_UBC_LIMIT);
}

static inline void pud_free (pud_t *pud)
@@ -74,7 +76,8 @@ static inline void pgd_list_del(pgd_t *p
static inline pgd_t *pgd_alloc(struct mm_struct *mm)
{
    unsigned boundary;
- pgd_t *pgd = (pgd_t *)__get_free_page(GFP_KERNEL|__GFP_REPEAT);
+ pgd_t *pgd = (pgd_t *)__get_free_page(GFP_KERNEL|__GFP_REPEAT|
+ __GFP_UBC | __GFP_UBC_LIMIT);
    if (!pgd)
        return NULL;
    pgd_list_add(pgd);
@@ -105,7 +108,8 @@ static inline pte_t *pte_alloc_one_kerne

static inline struct page *pte_alloc_one(struct mm_struct *mm, unsigned long address)
{
- void *p = (void *)__get_zeroed_page(GFP_KERNEL|__GFP_REPEAT);
+ void *p = (void *)__get_zeroed_page(GFP_KERNEL|__GFP_REPEAT|
+ __GFP_UBC | __GFP_UBC_LIMIT);
    if (!p)
        return NULL;
    return virt_to_page(p);
--- ./include/asm-x86_64/thread_info.h.ubslabs 2006-07-10 12:39:19.000000000 +0400
+++ ./include/asm-x86_64/thread_info.h 2006-08-01 15:20:30.000000000 +0400
@@ -78,14 +78,15 @@ static inline struct thread_info *stack_
    ({
        \
        struct thread_info *ret; \
        \
- ret = ((struct thread_info *)__get_free_pages(GFP_KERNEL,THREAD_ORDER)); \
+ ret = ((struct thread_info *)__get_free_pages(GFP_KERNEL_UBC, \
+ THREAD_ORDER)); \
        if (ret) \
            memset(ret, 0, THREAD_SIZE); \
        ret; \
    })
#else
#define alloc_thread_info(tsk) \
- ((struct thread_info *)__get_free_pages(GFP_KERNEL,THREAD_ORDER))
+ ((struct thread_info *)__get_free_pages(GFP_KERNEL_UBC,THREAD_ORDER))
#endif

#define free_thread_info(ti) free_pages((unsigned long) (ti), THREAD_ORDER)
--- ./ipc/msgutil.c.ubslabs 2006-04-21 11:59:36.000000000 +0400
+++ ./ipc/msgutil.c 2006-08-01 15:22:58.000000000 +0400
@@ -36,7 +36,7 @@ struct msg_msg *load_msg(const void __us
    if (alen > DATALEN_MSG)

```



```

alen = DATALEN_MSG;

- msg = (struct msg_msg *)kmalloc(sizeof(*msg) + alen, GFP_KERNEL);
+ msg = (struct msg_msg *)kmalloc(sizeof(*msg) + alen, GFP_KERNEL_UBC);
if (msg == NULL)
    return ERR_PTR(-ENOMEM);

@@ -57,7 +57,7 @@ struct msg_msg *load_msg(const void __us
    if (alen > DATALEN_SEG)
        alen = DATALEN_SEG;
    seg = (struct msg_msgseg *)kmalloc(sizeof(*seg) + alen,
-    GFP_KERNEL);
+    GFP_KERNEL_UBC);
    if (seg == NULL) {
        err = -ENOMEM;
        goto out_err;
--- ./ipc/sem.c.ubslabs 2006-07-10 12:39:19.000000000 +0400
+++ ./ipc/sem.c 2006-08-01 15:22:33.000000000 +0400
@@ -954,7 +954,7 @@ static inline int get_undo_list(struct s

    undo_list = current->sysvsem.undo_list;
    if (!undo_list) {
-    undo_list = kzalloc(sizeof(*undo_list), GFP_KERNEL);
+    undo_list = kzalloc(sizeof(*undo_list), GFP_KERNEL_UBC);
        if (undo_list == NULL)
            return -ENOMEM;
        spin_lock_init(&undo_list->lock);
@@ -1018,7 +1019,8 @@ static struct sem_undo *find_undo(int se
    ipc_rcu_getref(sma);
    sem_unlock(sma);

-    new = (struct sem_undo *) kmalloc(sizeof(struct sem_undo) + sizeof(short)*nsems,
GFP_KERNEL);
+    new = (struct sem_undo *) kmalloc(sizeof(struct sem_undo) +
+    sizeof(short)*nsems, GFP_KERNEL_UBC);
    if (!new) {
        ipc_lock_by_ptr(&sma->sem_perm);
        ipc_rcu_putref(sma);
@@ -1076,7 +1078,7 @@ asmlinkage long sys_sem timedop(int semid
    if (nsops > ns->sc_semopm)
        return -E2BIG;
    if (nsops > SEMOPM_FAST) {
-    sops = kmalloc(sizeof(*sops)*nsops, GFP_KERNEL);
+    sops = kmalloc(sizeof(*sops)*nsops, GFP_KERNEL_UBC);
        if (sops == NULL)
            return -ENOMEM;
    }
--- ./ipc/util.c.ubslabs 2006-07-10 12:39:19.000000000 +0400

```

```

+++ ./ipc/util.c 2006-08-01 15:18:45.000000000 +0400
@@ -302,9 +302,9 @@ void* ipc_alloc(int size)
{
    void* out;
    if(size > PAGE_SIZE)
-   out = vmalloc(size);
+   out = vmalloc_ub(size);
    else
-   out = kmalloc(size, GFP_KERNEL);
+   out = kmalloc(size, GFP_KERNEL_UBC);
    return out;
}

@@ -387,14 +387,14 @@ void* ipc_rcu_alloc(int size)
    * workqueue if necessary (for vmalloc).
    */
    if (rcu_use_vmalloc(size)) {
-   out = vmalloc(HDRLEN_VMALLOC + size);
+   out = vmalloc_ub(HDRLEN_VMALLOC + size);
    if (out) {
        out += HDRLEN_VMALLOC;
        container_of(out, struct ipc_rcu_hdr, data)->is_vmalloc = 1;
        container_of(out, struct ipc_rcu_hdr, data)->refcount = 1;
    }
    } else {
-   out = kmalloc(HDRLEN_KMALLOC + size, GFP_KERNEL);
+   out = kmalloc(HDRLEN_KMALLOC + size, GFP_KERNEL_UBC);
    if (out) {
        out += HDRLEN_KMALLOC;
        container_of(out, struct ipc_rcu_hdr, data)->is_vmalloc = 0;
--- ./kernel/fork.c.ubslabs 2006-07-31 18:40:20.000000000 +0400
+++ ./kernel/fork.c 2006-08-01 12:58:36.000000000 +0400
@@ -134,7 +134,7 @@ void __init fork_init(unsigned long memp
/* create a slab on which task_structs can be allocated */
task_struct_cachep =
    kmem_cache_create("task_struct", sizeof(struct task_struct),
-   ARCH_MIN_TASKALIGN, SLAB_PANIC, NULL, NULL);
+   ARCH_MIN_TASKALIGN, SLAB_PANIC | SLAB_UBC, NULL, NULL);
#ifdef
/*
@@ -1425,23 +1425,24 @@ void __init proc_caches_init(void)
{
    sighand_cachep = kmem_cache_create("sighand_cache",
        sizeof(struct sighand_struct), 0,
-   SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_DESTROY_BY_RCU,
+   SLAB_HWCACHE_ALIGN | SLAB_PANIC | \
+   SLAB_DESTROY_BY_RCU | SLAB_UBC,

```

```

    sighand_ctor, NULL);
    signal_cachep = kmem_cache_create("signal_cache",
        sizeof(struct signal_struct), 0,
-   SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL, NULL);
+   SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_UBC, NULL, NULL);
    files_cachep = kmem_cache_create("files_cache",
        sizeof(struct files_struct), 0,
-   SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL, NULL);
+   SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_UBC, NULL, NULL);
    fs_cachep = kmem_cache_create("fs_cache",
        sizeof(struct fs_struct), 0,
-   SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL, NULL);
+   SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_UBC, NULL, NULL);
    vm_area_cachep = kmem_cache_create("vm_area_struct",
        sizeof(struct vm_area_struct), 0,
-   SLAB_PANIC, NULL, NULL);
+   SLAB_PANIC|SLAB_UBC, NULL, NULL);
    mm_cachep = kmem_cache_create("mm_struct",
        sizeof(struct mm_struct), ARCH_MIN_MMSTRUCT_ALIGN,
-   SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL, NULL);
+   SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_UBC, NULL, NULL);
}

```

```

--- ./kernel/posix-timers.c.ubslabs 2006-04-21 11:59:36.000000000 +0400
+++ ./kernel/posix-timers.c 2006-08-01 12:58:57.000000000 +0400
@@ -242,7 +242,8 @@ static __init int init_posix_timers(void
    register_posix_clock(CLOCK_MONOTONIC, &clock_monotonic);

```

```

    posix_timers_cache = kmem_cache_create("posix_timers_cache",
-   sizeof (struct k_itimer), 0, 0, NULL, NULL);
+   sizeof (struct k_itimer), 0, SLAB_UBC,
+   NULL, NULL);
    idr_init(&posix_timers_id);
    return 0;
}

```

```

--- ./kernel/signal.c.ubslabs 2006-07-10 12:39:20.000000000 +0400
+++ ./kernel/signal.c 2006-08-01 12:59:14.000000000 +0400
@@ -2574,5 +2574,5 @@ void __init signals_init(void)

```

```

    kmem_cache_create("sigqueue",
        sizeof(struct sigqueue),
        __alignof__(struct sigqueue),
-   SLAB_PANIC, NULL, NULL);
+   SLAB_PANIC | SLAB_UBC, NULL, NULL);
}

```

```

--- ./kernel/user.c.ubslabs 2006-07-10 12:39:20.000000000 +0400
+++ ./kernel/user.c 2006-08-01 12:59:38.000000000 +0400
@@ -197,7 +197,7 @@ static int __init uid_cache_init(void)

```

```

int n;

uid_cachep = kmem_cache_create("uid_cache", sizeof(struct user_struct),
- 0, SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL, NULL);
+ 0, SLAB_HWCACHE_ALIGN|SLAB_PANIC|SLAB_UBC, NULL, NULL);

for(n = 0; n < UIDHASH_SZ; ++n)
    INIT_LIST_HEAD(uidhash_table + n);
--- ./mm/rmap.c.ubslabs 2006-07-10 12:39:20.000000000 +0400
+++ ./mm/rmap.c 2006-08-01 12:55:44.000000000 +0400
@@ -179,7 +179,8 @@ static void anon_vma_ctor(void *data, st
void __init anon_vma_init(void)
{
    anon_vma_cachep = kmem_cache_create("anon_vma", sizeof(struct anon_vma),
- 0, SLAB_DESTROY_BY_RCU|SLAB_PANIC, anon_vma_ctor, NULL);
+ 0, SLAB_DESTROY_BY_RCU|SLAB_PANIC|SLAB_UBC,
+ anon_vma_ctor, NULL);
}

/*
--- ./mm/shmem.c.ubslabs 2006-07-10 12:39:20.000000000 +0400
+++ ./mm/shmem.c 2006-08-01 13:26:13.000000000 +0400
@@ -367,7 +367,8 @@ static swp_entry_t *shmem_swp_alloc(stru
}

spin_unlock(&info->lock);
- page = shmem_dir_alloc(mapping_gfp_mask(inode->i_mapping) | __GFP_ZERO);
+ page = shmem_dir_alloc(mapping_gfp_mask(inode->i_mapping) | \
+ __GFP_ZERO | __GFP_UBC);
if (page)
    set_page_private(page, 0);
spin_lock(&info->lock);

```
